UNIVERZA V MARIBORU

FAKULTETA ZA ELEKTROTEHNIKO,

RAČUNALNIŠTVO IN INFORMATIKO

Domen Hribernik

CONWAYEVA IGRA ŽIVLJENJA

Projektna naloga

Maribor, april 2024

KAZALO

KAZALO SLIK

UPORABLJENI SIMBOLI IN KRATICE

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

JS – JavaScript

JSON – JavaScript Object Notation

$O(n^2)$ – kvadratna časovna zahtevnost

Conwayeva igra življenja

Ključne besede: Conwayeva igra življenja, implementacija

Povzetek

Raziskovalna naloga preučuje Conwayevo igro življenja. Teoretični del obravnava pravila, osnovne objekte, zgodovino igre in kompleksnost igre. Praktični del vključuje lastno implementacijo igre. Cilj je analizirati in razumeti igro ter predstaviti lastno implementacijo.

Conway's game of life

Keywords: Conway's game of life, implementation

Abstract

Research paper examines Conway's Game of Life. The theoretical part addresses rules, basic objects, history of the game, and complexity of the game. The practical section includes a custom implementation of the game. The goal is to analyze and understand the game and present a custom implementation.

1 Uvod

Conwayeva igra življenja, ki jo je leta 1970 ustvaril britanski matematik John Conway, predstavlja eno najbolj znanih primerov celularnih avtomatov. Igra deluje na preprostih pravilih o evoluciji populacije celic v dvodimenzionalnem okolju, ki zahteva le osnovno znanost o sistemih za njeno razumevanje.

V tem projektu se bom osredotočil na raziskovanje in razumevanje konceptov Conwayeve igre življenja ter implementacijo lastne različice igre. Moj cilj je raziskati in implementirati to igro tako, da bo njena implementacija optimizirana. Želim tudi narediti grafični vmesnik, ki prikaže delovanje te igre.

V sklopu projekta bom razvil svojo implementacijo igre. S pomočjo katere bom prikazal, kako deluje ta igre v omejenem območju, po korakih ali z animacijo.

V nadaljevanju bom podrobneje predstavili osnove Conwayeve igre življenja, opisali osnovne objekte igre, predstavili zgodovino igre. Za tem bom razkrili zanimive lastnosti igre ter razmišljal o praktični uporabi igre na različnih področjih.

2 Teoretični del

2.1 Zgodovina

Ena izmed najbolj znanih celic avtomatov je Conwayeva igra življenja, ki jo je ustvaril britanski matematik John Horton Conway v začetku 70. let prejšnjega stoletja. Kljub svoji enostavnosti se igra odlikuje po zapletenem in nepredvidljivem karakterju, kar ji je zagotovilo veliko pozornosti.

Leta 1970 je bil prvič predstavljen Conwayev »Game of Life« v reviji Scientific American. Igra uporablja preproste celice avtomata in se osredotoča na nekaj enostavnih pravil, ki kljub svoji preprostosti omogočajo nastanek izjemno zapletenih vzorcev in obnašanja.

Po objavi v reviji je Conwayeva igra življenja hitro postala vir zanimanja za računalničarje, matematike in ljubitelje iger. Njena popularnost je privedla do številnih raziskav in študij, ki so razkrili različne vidike igre ter pripomogli k razvoju novih matematičnih in računalniških konceptov.

Med pomembnimi mejniki v zgodovini Conwayeve igre življenja je bila odkritje stabilnih formacij, različnih vzorcev in periodičnih struktur. Ti dosežki so pripomogli k razumevanju kompleksnosti igre ter njenega potenciala za uporabo v različnih aplikacijah, vključno z biologijo, računalništvom in umetno inteligenco.

2.2 Pravila igre

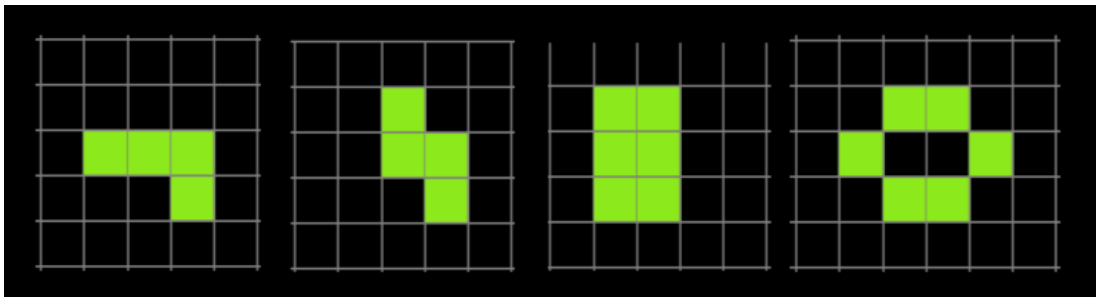
V igri življenja je vesolje, neskončna, dvodimenzionalna pravokotna mreža kvadratnih celic, pri čemer je vsaka celica v enem izmed dveh možnih stanj mrtva ali živa (0 ali 1). Vsaka celica se odziva s svojimi osmimi sosednjimi celicami, ki so vodoravno, navpično ali diagonalno sosednje. Pri vsakem koraku v času pride do naslednjih prehodov:

- Katera koli živa celica z manj kot dvema živima sosedoma umre, zaradi »premajhne poselitve«.
- Katera koli živa celica z dvema ali tremi živima sosedoma preživi v naslednjo generacijo.

- Katera koli živa celica z več kot tremi živimi sosedi umre, kot da bi bila »prenaseljena«.
- Katera koli mrtva celica s točno tremi živimi sosedi postane živa celica, kot da bi se »razmnožila«.

Začetni vzorec predstavlja seme sistema. Prva generacija se ustvari z uporabo zgoraj navedenih pravil, tako da nekatere celice umrejo, živijo naprej ali pa se razmnožijo. Rojstva in smrti se zgodijo hkrati v diskretnem trenutku, ki ga imenujemo korak. Vsaka generacija je čista funkcija prejšnje. Pravila se nadaljujejo z uporabo, da se ustvarijo nadaljnje generacije.

Na spodnji sliki predstavljam primer pravil igre, kjer lahko vidimo potek od leve proti desni.



Slika 1 - Pravila igre

Na sliki imamo štiri korake, ki prikazujejo razvoj igre skozi čas. V prvem koraku umre prvi kvadratek na koordinatah (2, 3), saj ima samo enega soseda, medtem ko se v drugem kvadrat rodi celica (3, 2), saj ima v prejšnjem koraku točno tri sosede. Podobno vidimo v naslednjih korakih, dokler ne pridemo do zadnje mreže, kjer nastane ena izmed nespremenljivih formacij. Vsaka celica ima dva ali tri sosede, tako da bo ta struktura ostala enaka do konca simulacije.

S temi preprostimi štirimi pravili, pa se da narediti marsikaj, tega se bom dotaknil v naslednjih poglavjih, kar naredi to igro zelo zanimivo. Potrebno je pa omeniti, da je ta igra lahko implementirana neskončno ali pa končno. Če je igra končna to pomeni, da je polje omejeno na fiksno širini in višino in se celice drugače obnašajo na robu igralnega polja.

2.3 Preprosti objekti igre

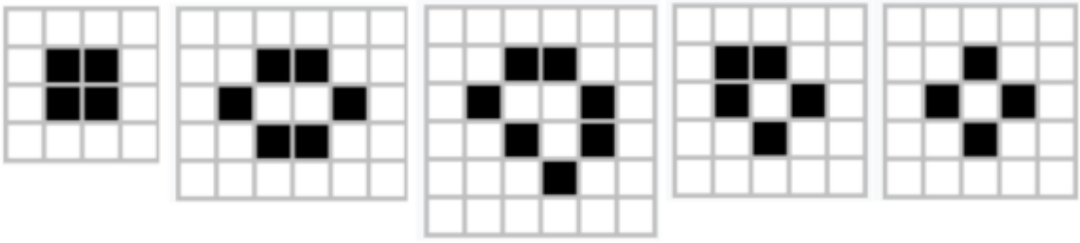
V igri življenja pride do zanimivega pojava več celic, lahko skupaj veliko različnih vrst vzorcev. Če je igra neskončna, je tudi teh vzorcev neskončno, ampak lahko jih pa razdelimo v tri osnovne skupine. [1]

- Nespremenljivke (Still lifes): Vzorci, ki se ne spreminjajo iz enega koraka v drugega. Tako, da imajo vse celice dva ali tri sosede.
- Oscilatorji (Oscillators): Vzorci, ki se po določenem številu korakov vrnejo v začetno stanje.
- Vesoljske ladje (Spaceships): Vzorci, ki se premikajo po mreži v določeno smer in ne izumrejo.

Najzgodnejši zanimivi vzorci v igri življenja so bili odkriti brez uporabe računalnikov. Najpreprostejše nespremenljivke in oscilatorje so odkrili med sledenjem usod različnih majhnih začetnih konfiguracij z uporabo tabel ali fizičnih igralnih plošč kot za igro Go.

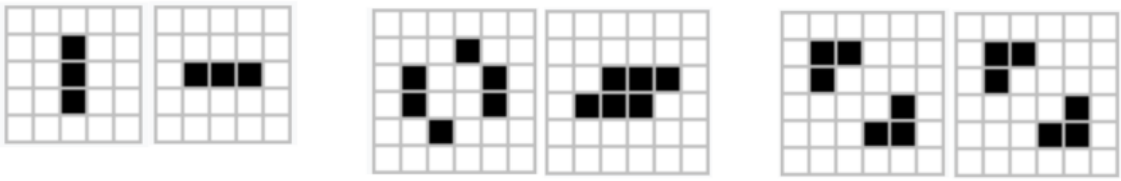
Spodaj so prikazani pogosto pojavljajoči se primeri treh zgoraj navedenih vrst vzorcev (v smislu da se pogosto pojavljajo iz naključne, začetne konfiguracije celic), pri čemer so žive celice prikazane črno, mrtve pa belo. Obdobje se nanaša na število korakov, skozi katere mora vzorec iterirati, preden se vrne v svojo začetno konfiguracijo.

Na spodnji sliki so vse nespremenljivke, vsaka ima tudi specifično ime.



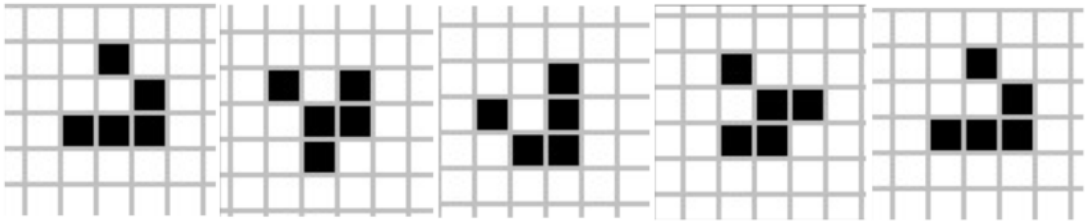
Slika 2 – Nespremenljivke

Pri naslednji sliki lahko vidimo kako delujejo trije najpreprostejši oscilatorji.



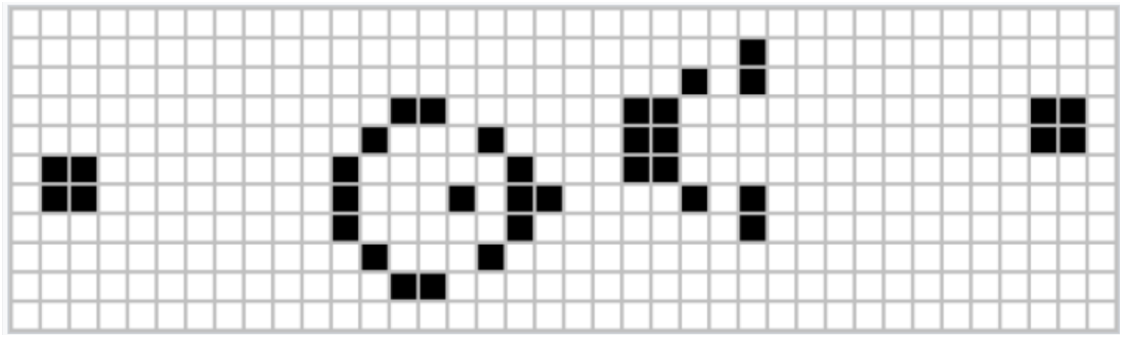
Slika 3 - Oscilatorji

Na zadnji sliki je primer najpreprostejše vesoljske ladje, ki je tudi najbolj uporabna. Lahko vidimo, kako se v petih korakih vrne v privotno stanje, le da med tem potuje v spodnje desno diagonalno smer.



Slika 4 - Vesoljska ladja

Obstajajo tudi kompleksnejši objekti, kot na primer generator, ampak ta in vsi ostali objekti so zglajeni iz osnovnih. Na primer generator je zgrajen iz oscilatorjev in vesoljskih ladij, deluje pa tako, da interakcija dveh ali oscilatorjev in nespremenljivk za proizvajanje neskončnih vesoljskih ladij, ki letijo v neskončnost. Na spodnji sliki vidimo Gosperjev generator, ki je eden prvih generatorjev. [2]

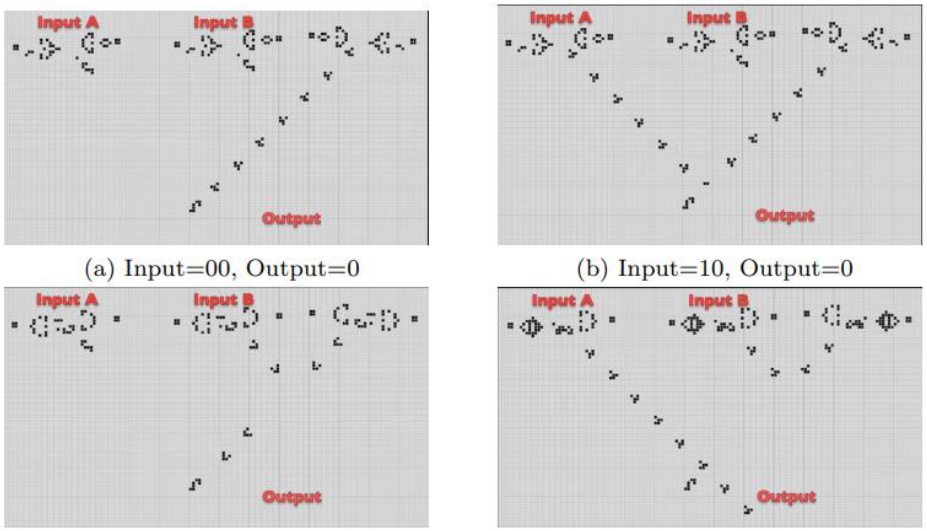


Slika 5 - Gosperjev generator vesoljskih ladij

2.4 Kompleksnost igre

Kot večkrat omenjeno, je Conwayeva igra življenja zelo preprosta, ima samo štiri pravila, ampak lahko iz teh pravil naredimo zelo kompleksne sisteme.

V prejšnjem poglavju sem pokazal tri osnovne tipe objektov in s pomočjo teh lahko naredimo marsikaj. Če gremo postopoma, je najpomembnejša stvar Gosperjev generator vesoljskih ladij, ki je prikazan na koncu prejšnjega poglavja. S pomočjo več teh generatorjev in nekaj drugih objektov, ki uničujejo ladje, lahko naredimo preprosto logiko. Na spodnji sliki je prikazano, kako lahko naredimo IN vrata s pomočjo treh generatorjev. [3]



Slika 6 - IN vrata

Na podoben način lahko naredimo ALI vrata in tudi vrata za negacijo in s pomočjo teh objektov imamo osnovno logiko in lahko naredimo, karkoli želimo. Lahko naredimo digitalna vezja in kompleksne programe. Najbolj zanimivo pa je, da lahko sprogramiramo igro življenja samo v sebi. [4] [5]

2.5 Uporaba Conwayeve igre življenja

Conwayeva igra življenja ima kljub preprosti naravi različne aplikacije v različnih področjih, na primer v biologiji, računalništvu in umetni inteligenci.

- Biologija: Igra življenja, se uporablja kot model za raziskovanje bioloških sistemov, kot so širjenje bolezni, populacijska dinamika in evolucijski procesi.

Matematični vzorci, ki jih generira igra življenja, lahko ponujajo vpogled v kompleksnost nekaterih bioloških sistemov in omogočajo simulacije za testiranje hipotez pri omenjenih primerih.

- Računalništvo: V računalništvu se igra življenja pogosto uporablja za preizkušanje in demonstracijo algoritmov, ki se ukvarjajo z iskanjem vzorcev, avtomatsko generacijo vzorcev in distribuiranim računalništvom. Poleg tega je igra življenja tudi vir navdiha za razvoj celo vrsto računalniških sistemov.
- Umetna inteligenca: Igra življenja, se pogosto uporablja za preučevanje evolucijskih algoritmov in samo učenja. Raziskovalci uporabljajo igro življenja kot model za razvoj inteligentnih sistemov, ki se lahko prilagajajo in optimizirajo glede na okolje.

Poleg teh področij ima igra življenja tudi različne druge aplikacije, vključno z vzorčenjem in digitalno umetnostjo. Njena preprostost omogočata raznovrstne načine uporabe v številnih disciplinah.

3 Metodologija

Cilj raziskave je analizirati in razumeti Conwayevo igro življenja ter izdelati lastno implementacijo.

Pristop obsega študij literature o pravilih igre, zgodovini in osnovnih objektih, kar služi razumevanju igre. Za implementacijo se uporablja programski jezik JS z uporabo HTML in CSS za vizualizacijo. Ta orodja so izbrana z namenom preproste in hitre implementacije ter razširljivosti, saj bo končna aplikacija podprta na namiznih in mobilnih napravah.

4 Rezultati

4.1 Opis implementacije logike

Na začetku sem implementiral vodič za uporabnika, ta bo prikazan na zgornjem levem kotu in uporabniku pokazal, kako uporabljati program.

Nato sem implementiral igro. Ustvaril sem nadzornika, ki čaka na pritisk tipk D, S in Q. Ob pritisku S se začne animacija, ob Q se ustavi animacija in ob D se izvede samo en korak. Na spodnji sliki je prikazana koda delovanja nadzornika.

```
let gameController = e => {
  e.key == "d" ? gameOfLife() :
  e.key == "s" && !x ? x = setInterval(gameOfLife, 100) :
  e.key == "q" ? (clearInterval(x), x = null) : null;
}

document.addEventListener('keypress', gameController);
```

Slika 7 - Nadzornik igre

Zdaj sem implementiral igro, to sem storil v funkciji gameOfLife(). To funkcijo tudi kliče nadzornik če hočemo en korak, jo kliče enkrat, če pa želimo animacijo, pa kličemo funkcijo desetkrat na sekundo. Ta interval tudi počistimo, ko uporabnik želi ustaviti igro.

Preden lahko izvajamo igro, jo moramo ustvariti to naredim tako, da definiram konstante, ki so grafični element igre višina, širina in matriko, ki vsebuje podatke igre. To prikazujem na spodnji sliki.

```
const game = document.getElementById("game");
let arr = [];
let x = null;
const height = 30;
const width = 70;
```

Slika 8 - Inicializacija igre

Spremenljivka x služi shranjevanju intervala igre.

Sedaj lahko ustvarim polje na grafičnem vmesniku. To storim tako, da z dvojno zanko narišem vsak element na ekran in z drugo dvojno zanko ustvarim matriko igre. To prikazujem na spodnji sliki. Matrika je večja za dve mesti, saj se tako izognem napaki, ko gre objekt izven vidnega polja.

```
for (let i = 0; i < height; i++) {
  for (let j = 0; j < width; j++) {
    let cell = document.createElement("div");
    cell.classList.add("cell", "h"+(i+1), "w"+(j+1));
    cell.addEventListener('click', addCell);
    game.append(cell);
  }
}

for (let i = 0; i < height+2; i++) {
  arr.push([]);
  for (let j = 0; j < width+2; j++) {
    arr[0].push(0);
  }
}
```

Slika 9 - Kreiranje mreže

Zdaj lahko implementiram metodo gameOfLife(), ki jo kliče nadzornik. Ta funkcija pridobi podatke iz matrike, ki je shranjena v JSON formatu. Nato z dvojno zanko preletimo celotno polje in za vsako celico izračunamo njene sosede. To naredimo s pomočjo funkcije getNeighbours(i, j), ki vzame koordinate celice in vrne število sosedov to je prikazano na spodnji sliki.

```
let getNeighbours = (x, y) => {
  let sum = 0;
  for (let i = x-1; i < x+2; i++) {
    for (let j = y-1; j < y+2; j++) {
      if (arr[i][j] == 1 && (x != i || y != j)) {
        sum++;
      }
    }
  }
  return sum;
}
```

Slika 10 - Pridobivanje števila sosedov

Zdaj Lahko naredimo glavno logiko igre. Če je v trenutni celici število 1, kar pomeni, da je trenutna celica v živem stanju, preverimo, ali ima več kot tri sosede ali manj kot dva

soseda, in v primeru, da je to res, nastavimo njeno stanje na 0, kar pomeni, da je celica umrla. Če pa ima celica stanje 0, pa preverimo, ali ima točno 3 sosede, in v tem primeru nastavimo njeno stanje na 1, kar pomeni, da se je celica razmnožila. Tako smo štiri pogoje igre implementirali v le dveh pogojnih stavkih, kar je optimiziralo program. Na spodnji sliki prikazujem glavno funkcijo programa.

```
let gameOfLife = () => {
  const tmp = JSON.parse(JSON.stringify(arr));
  let neighbours;
  for (let i = 1; i <= height; i++) {
    for (let j = 1; j <= width; j++) {
      neighbours = getNeighbours(i, j)
      arr[i][j] == 1
      ? neighbours > 3 || neighbours < 2
        ? tmp[i][j] = 0 : null
      : neighbours == 3 ? tmp[i][j] = 1 : null;
    }
  }
  arr =  JSON.parse(JSON.stringify(tmp));
  colorGrid();
}
```

Slika 11 - Glavna funkcija igre

Na koncu je še potrebno shraniti podatke v format JSON in spremembe prikazati na grafični vmesnik. Prikaz na grafični vmesnik naredimo s pomočjo funkcije colorGrid(), ki je implementirana tako, da z dvojno zanko preleti vsako celico in ji doda ustrezen razred. Če ima matrika stanje 1, potem doda celici razred »life«, v nasprotnem primeru, če ima celica stanje 0, potem preverimo, ali ima razred »life« in če ga ima, ji ga vzamemo. Ta razred je obdelan v CSS tako, da so prikazane samo celice, ki imajo življenje. Na spodnji sliki prikazujem funkcijo, ki opravlja izris.

```
let colorGrid = () => {
  let elem;
  for (let i = 1; i <= height; i++) {
    for (let j = 1; j <= width; j++) {
      elem = document.getElementsByClassName(`h${i} w${j}`)[0];
      arr[i][j] == 1 ? elem.classList.add("life") : null;
      arr[i][j] == 0 && elem.classList.contains("life") ? elem.classList.remove("life") : null;
    }
  }
}
```

Slika 12 - Opravljanje izrisa

4.2 Stil igre

Stil sem v celoti naredil v CSS in prikazujem glavni del (odsek, ki ima največ pomena) na spodnji sliki.

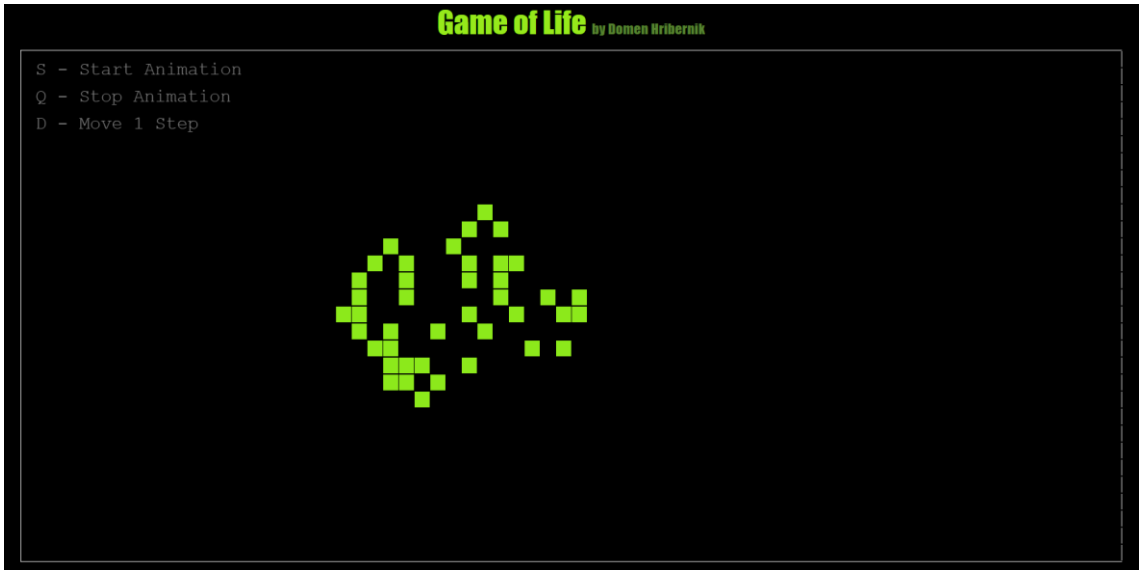
```css
#game {
  display: inline-grid;
  border: 1px solid white;
  grid-template-columns: repeat(70, 25px);
  margin-top: 15px;
}

.title, #game {
  position: relative;
  left: 50%;
  transform: translateX(-50%);
}

.cell {
  display: inline-block;
  height: 25px;
  width: 25px;
  border: 1px solid black;
}

.life {
  background: rgb(140, 233, 27);
}
```

Slika 13 - Stil igre

Na sliki lahko vidimo, kako je narejena glavna igra, to je z grid elementom. Celotna mreža je poravnana na sredino ekrana, celice so pobarvane na črno, razen če imajo razred »life«, ki jim da svetlo zeleno barvo.

4.3 Grafični vmesnik

Na koncu je grafični vmesnik zelo preprost. Ima glavni naslov in podnaslov, ki je moje ime kot avtor programa. Nato je na celem oknu prikazana mreža, ki ima v zgornjem kotu vodič za uporabnika, in v mreži se dogaja igra (samo en korak na sliki).



Slika 14 - Grafični vmesnik

4.4 Analiza učinkovitosti

Večinoma se program izvaja z $O(n^2)$ kompleksnostjo. Saj imamo matriko in v njej moramo preveriti ter posodobiti vsako celico. To je slabo optimizirano, saj se igra ne dogaja na celotni mreži. Ampak glede na to, da se funkcija kliče enkrat na 100ms, kar je zelo počasi, in da je polje omejeno na majhno mrežo, je ta implementacija zadostna.

Optimizacijo sem pa naredil pri pogoju igre, kot sem omenil v prvem podpoglavju, kjer sem vsa pravila uspel povzeti v samo dveh pogojih. To sta pogoj za umiranje in pogoj za rojstvo, saj sta to edina pogoja, ki dejansko spreminjata grafični vmesnik. Pogoj obstoja bi lahko bil zanimiv, če bi nas zanimalo, koliko časa je obstal element, ali pa če bi skrbeli, kako daleč se širi igra, ampak to me ne zanima v moji implementaciji.

5 Diskusija

5.1 Povzetek Doseženih Ciljev

- Preučili sem in razumeli osnovne koncepte Conwayeve igre življenja.
- Uspešno sem implementirali lastno različico igre življenja v jeziku JS.
- Razvil sem grafični vmesnik, ki omogoča interaktivno uporabo implementirane igre.

5.2 Evalvacija Rezultatov

- Implementacija je omogočila uspešno simulacijo in vizualizacijo različnih scenarijev igre življenja.
- Grafični vmesnik je intuitiven za uporabo in je izboljšal uporabniško izkušnjo.
- Implementacija igre na se je pokazala manj optimizirana, kot sem pričakoval na začetku, ampak je igra ustrezna zaradi počasnega izvajanja (stilska odločitev) in omejenega območja igre.

5.3 Odgovori na Raziskovalna Vprašanja

- Uspešno sem implementiral svojo verzijo Conwayeve igre življenja.
- Identificirali sem ključne dejavnike, ki vplivajo na optimizacijo igre, in nekatere tudi implementiral.

5.4 Sklepna Misel

Rezultati moje implementacije in analize kažejo na uspešno dosežene cilje ter ponujajo vpogled v dinamiko in kompleksnost Conwayeve igre življenja. Sam sem tudi zadovoljen s svojo implementacijo, saj zavzema malo vrstic kode in menim, da ima dober grafični vmesnik.

6 Zaključek

Cilj raziskave je bil razumeti in implementirati Conwayevo igro življenja ter implementirati optimizirano rešitev.

Raziskava in implementacija igre življenja sta pomembni za razumevanje dinamike kompleksnih sistemov ter za razvoj in optimizacijo algoritmov. Rezultati ponujajo osnovo za implementacijo bolj optimizirane igre.

Prihodnji koraki vključujejo razširitev implementacije z dodajanjem naprednih funkcionalnosti, kot so različni načini uporabe pravil. Dodatno bi lahko razvili tudi bolj kompleksne modele igre življenja, ki vključujejo dodatne elemente in pravila, kot na primer igra življenja, kjer so celice šest kotniki. Moja naloga dobro osnovo za nadaljnje implementacije te igre.

7 Literatura in viri

[1] N. Johnston in D. Greene, Conway's Game of Life: Mathematics and Construction, Morrisville: Lulu.com, 2021.

[2] P. Callahan, „Paul's Page of Conway's Life Miscellany," 10 marec 2019. [Elektronski]. Available: https://conwaylife.com/ref/lifepage/. [Poskus dostopa 20 april 2024].

[3] G. Yuncong, *The Game of Life on the Hyperbolic Plane,* Terre Haute: Rose-Hulman Institute of Technology, 2020.

[4] P. Bradbury, „Life in life," 13 maj 2012. [Elektronski]. Available: https://www.youtube.com/watch?v=xP5-iIeKXE8. [Poskus dostopa 18 april 2024].

[5] D. Beattie, „The Art of Code," 26 februar 2020. [Elektronski]. Available: https://www.youtube.com/watch?v=6avJHaC3C2U&t=603s. [Poskus dostopa 19 april 2024].